LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-410145

Feasibility of N-Gram Data-Structures for Next-Generation Pathogen Signature Design

S. N. Gardner

January 27, 2009

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

FY08 LDRD Final report
Feasibility of N-Gram Data-Structures for Next-Generation Pathogen Signature Design
LDRD Project Tracking Code: 08-FS-014
Shea N. Gardner, Principal Investigatory
January 26, 2009
Lawrence Livermore National Laboratory

We determined the most appropriate data structure for handling n-gram (also known as k-mer) string comparisons and storage for genomic sequence data that will scale in terms of memory and speed. This is critical to maintain LLNL as the leader in pathogen detection, as it will guide the design of the "Next Generation" system for computational signature prediction.

There are two parts to k-mer analysis for signature prediction that we investigated. First is the enumeration and frequency counting of all observed k-mers in a sequence database (k-mer is a biological term equivalent to the CS term n-gram). Second is the down-selection and pairing of k-mers to generate a signature. We determined that for the first part, suffix arrays are the preferred method to enumerate k-mers, being memory efficient and relatively easy and fast to compute. For the second part, a subset of the k-mers can be stored and manipulated in a hash, that subset determination based on desired frequency characteristics such as most/least frequent from a set, shared among sequence sets, or discriminating across sequence sets.

The first step of computing all the observed k-mers and their frequencies in a sequence database poses the memory bottleneck. Initially, we investigated hashing functions for turning data into a relatively small integer to serve as an index into an array, to speed up k-mer searches and counts for each k-mer in the sequence database and eliminating duplicates, figuring that actual k-mer sequences could be stored in flatfiles without demanding RAM. Conceptually, hashes are appealing, since they can be used as a dictionary. For a given k-mer stored as the key, any number of alternative data types can be linked as values, such as the genomes, species, or genes containing that k-mer, facilitating downstream analysis linking k-mers to taxonomy or function. This may be fruitful for investigations of advanced/synthetic/engineered biothreats or analysis of metagenomic or high-coverage, short-read sequence data (e.g. from Illumina sequencing). Practically, however, we were unable to find a hashing function that met our needs. Since a hashing function may map several different keys to the same hash value, we found that once the hash table fills to approximately 25% of available space, the number of collisions increases, substantially slowing performance to build the hash. Thus, extremely large memory had to be set aside in advance so that less than 25% of it would be filled, which was prohibitive for all bacterial genomes. While perfect hashing algorithms can handle key sets with billions of keys, which we hoped could improve performance, further reading indicated that they are complicated and less efficient for dynamic sets (as compared with static sets) as is required to build the hash initially, that is, the process of scanning all the sequences in the database and adding k-mers to the hash as new ones are observed. Nor is cuckoo hashing suitable, since it is possible that a given k-mer can be indexed more than once in the hash. Finally, chain hashing was eliminated as it requires an extra 64 bits of memory for a pointer, countering our aim for

maximum memory conservation. Thus, we turned to a different approach that did not require hashing, namely, suffix arrays.

We found that suffix arrays to compute all observed k-mers and their frequency counts a fast, memory-efficient method for determining k-mer frequencies in large sequence databases. The memory required for this computation is a function of the database size, approximately ~33 bytes per DNA base. We used existing suffix array open source code from http://www.cs.dartmouth.edu/~doug/sarray/ , modifying to handle long, unsigned int and developing Python scripts for downstream processing of the output such as determining the N most frequent or infrequent k-mers in a set of sequences. We can parallelize k-mer frequency counts for data sets too large to fit in RAM by partitioning the sequence data and then performing an out-of-core merge and sort on the results from each partition. While this makes the approach scaleable even for data sets that exceed available RAM, unfortunately, the out-of-core merge sort is slow, so large memory machines will enable larger datasets to be handled much faster without partitioning. Currently, sequence datasets over 4 GB must be partitioned in order to be run on a 32 GB node. Our all_bacteria database containing all available bacterial complete genome and plasmids was over 5 GB when we did a test run (it is now > 6 GB), and it required ~12 hours to do the suffix array and merge sort on the 17 partitions with k=20. Longer k required more time (e.g. k=60 required ~17 hours). We determined that in all viral genomes, we observe about 35% of the number of k-mers one would expect to see in a random sequence database of that size for k between 20 and 60. Similarly, for bacteria, one observes 45% of the expected number.

We provided test code to Maya Gokhale, Dave Fox, Eric Greenwade, and a collaborator at U. Houston for benchmarking memory requirements on different machines, including SiCortex, Violin flash-disk system, and a 500 GB SGI. They determined that the SGI had a 3x speed penalty for having to access disk over their fabric compared to the 256 GB RAM from MetaRam, a quad socket quad core Opteron, with 64 GB RAM that is "local" to each of the 4 sockets. These benchmarks were useful in LLNL evaluation of various of large memory systems, and contributed to the decision of which large memory nodes LLNL would purchase.

We successfully concluded this project, determining which data structures would be most useful for k-mer computations. Suffix arrays will be used for k-mer enumeration and sorting, and hashes will be used downstream for postprocessing (e.g. for signature design) on subsets of those k-mers selected based on their frequencies. Code has been developed to perform suffix array computations. We worked with the storage-intensive super computing team, providing them with benchmarking code to run on different machines for comparison, the results of which figured into decisions as to which large memory computers LLNL purchased. The results of this study are now being applied to Intelligence Technology Innovation Center/Office of the Chief Scientist sponsored work on emerging/engineered/advanced threat detection and to the the LDRD SI "Viral Discovery Platform". David Hysom and Peter L. Williams prototyped code for this project, and Shea Gardner oversaw the project.

Web summary

For the k-mer (n-gram) analysis, suffix arrays are preferred, more memory efficient and faster than other (e.g. various hashing) methods. A subset of the k-mers

selected based on frequency characteristics can be stored and manipulated in a hash for downstream tasks such as signature prediction or functional correlations.

We provided test code to storage-intensive super computing collaborators for benchmarking on different machines, including SiCortex, Violin flash-disk system, and a 500 GB SGI. These benchmarks were useful in LLNL evaluation of various large memory systems, and contributed to the decision of which large memory nodes LLNL would purchase.

Auspices Statement

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

This work was funded by the Laboratory Directed Research and Development Program at LLNL under project tracking code 08-FS-014.